7/23/03

REMARKS

Claims 1-37 are pending in the present application. Claims 2, 13, 24, and 35 were canceled; claims 1, 12, 23, 28, and 34 were amended. Reconsideration of the claims is respectfully requested.

Claim 28 was amended to depend from claim 27, thereby providing the proper antecedent basis for the limitation, "the system", as recited in claim 28.

I. 35 U.S.C. § 102, Anticipation, Claims 1-4, 8, 9, 1-15, 19, 20, 22-26, 30, 31, 33, 34, 36, and 37

The Examiner has rejected claims 1-4, 8, 9, 1-15, 19, 20, 22-26, 30, 31, 33, 34, 36, and 37 under 35 U.S.C. § 102 as being anticipated by *Gerra et al.* (US Patent No. 6,229,534). This rejection is respectfully traversed.

With regard to claims 1, 12, and 23, the Examiner states:

> As per claims 1, 12, and 23 Gerra discloses a method, computer program code with instructions for, and a system with means for, in a data processing system, comprising the steps of:
> receiving from a client, a request for a host screen (column 8, lines 1-10; column 9, lines 11-37);
> navigating to the host screen (column 8, lines 9-23; column 9, lines 11-37)
> retrieving the host screen (column 8, lines 9-43, line 51-67; column9, lines 1-52);
> formatting the host screen into a formatted host screen (column 9, lines 37-52; column 10, lines 1-13, lines 23-30); and
> sending the formatted host screen to the client (column 9, lines 37-52; column 10, lines 1-13, lines 23-30).

(*Office Action*, dated April 23, 2003, page 3).

A prior art reference anticipates the claimed invention under 35 U.S.C. §102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). The *Gerra* reference cited by the Examiner does not anticipate the present invention as recited in claims 1, 12, and 23, because *Gerra* fails to teach each and every element of the claims. Amended independent claim 1, which is representative of amended independent claims 12 and 23, reads as follows:

1. A method in a data processing system, comprising the steps of:
receiving, from a client, a request for a host screen;
navigating to the host screen;
retrieving the host screen;
formatting the host screen into a formatted host screen, wherein the formatted host screen displays selectable links to other screens within a host system; and
sending the formatted host screen to the client.

Gerra fails to teach formatting the host screen into a formatted host screen, wherein the formatted host screen displays selectable links to other screens within a host system, as recited in amended claim 1. The Examiner refers to the following passage as evidence that *Gerra* teaches this feature:

> Those skilled in the art will recognize that browser 220 may generate additional frames to display web pages and corresponding BIN buttons, if any, for additional servers. Alternatively, an aggregated web page and corresponding BIN buttons, if any, for all servers may be displayed in a single frame. In other words, although the figures show two frames, the first for the legacy application and the second for an HTML-based application, the present invention contemplates multiple frames, one for each legacy application and HTML-based application. Alternatively, a single frame may be used to display aggregated information from multiple legacy applications and a single frame may be used to display an aggregated web page from multiple HTML-based applications.

(*Gerra*, col. 9, line 67 to col. 10, line 13). The passage above teaches, when interfacing information from at least two remote sources (e.g. a legacy application and an internet-based application), the legacy application can be displayed in a single or in multiple frames, and the internet-based application can likewise be displayed in a single or in multiple frames. Figure 8 in *Gerra* provides an example of a multi-frame browser display.

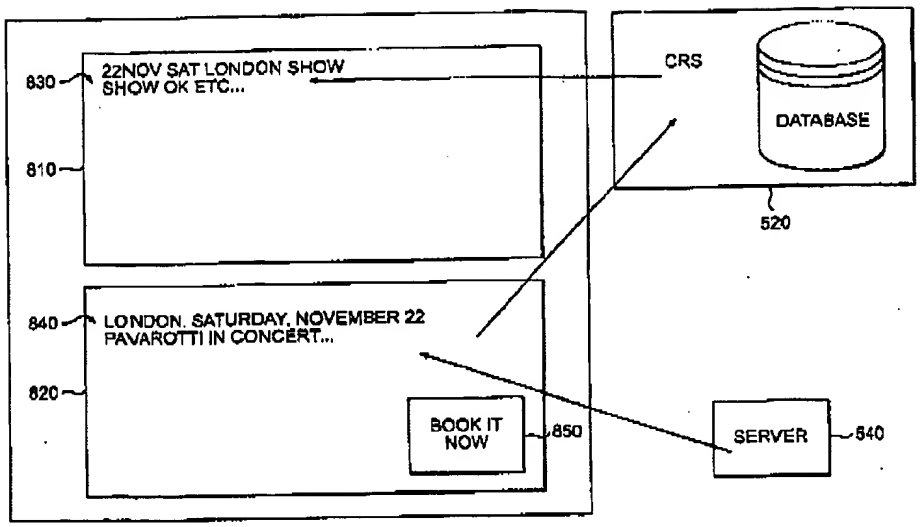

FIG. 8

Figure 8 above illustrates how the multiframe browser is displayed, with the legacy application contained in one frame and the remote source internet-based application including a "book-it-now" button contained in another frame. Although *Gerra* discloses the feature of including additional legacy applications and/or addition internet-based applications within the browser, there is no mention of having the formatted host screen (legacy application) display selectable links to other screens within the host system. To the contrary, the *Gerra* reference merely teaches having multiple legacy applications displayed within a single frame and/or having multiple internet-based applications within another single frame. The legacy application and the internet-based application are not displayed in the same frame. In view of the passage above and Figure 8, the "book-it-now" button resides on the internet-based application frame. No matter how many additional frames are added to display web pages (e.g., to provide additional information about London or Pavarotti) or legacy applications, the "book-it-now" button of the remote source internet-based application teaches sending information to the legacy application, rather than have the Legacy application display a selectable link to other screens within the host system, as recited in amended claim 1. Thus, *Gerra* fails to teach having the formatted host screen (legacy application) display selectable links to other

screens, not to mention displaying selectable links to other screens within the host system as recited in amended claim 1.

In view of the above, *Gerra* fails to teach each and every element of amended claim 1. Providing the internet-based application with a "book-it-now" button to transfer information to the legacy application in *Gerra* does not teach the present invention's feature of having the formatted host screen display selectable links to other screens within the host system. As a result, the *Gerra* reference fails to anticipate amended claims 1, 12, and 23 of the present invention.

With regard to claim 34, the Examiner states:

> As per claim 34, *Gerra* discloses a macro bean for providing navigation between screens within a legacy host system, the macro bean comprising:
> first instructions for receiving a request for a requested host screen from a legacy host system (column 8, lines 1-10, lines 24-43; column 9, lines 11-37);
> second instructions for determining the current host screen (column 8, lines 1-22, 51-67; column 9, lines 11-37);
> third instructions for navigating to the requested host screen (column 8, lines 9-23; column 9, lines 11-37).

(*Office Action*, dated April 23, 2003, page 3). Amended independent claim 34 reads as follows:

> 34. A macro bean for providing navigation between screens within a legacy host system, the macro bean comprising:
> first instructions for receiving a request for a requested host screen from a legacy host system;
> second instructions for determining the current host screen; and
> third instructions for navigating to the requested host screen, wherein intermediate host screens between the current host screen and the requested host screen are unsent to a client.

Claim 34 recites the feature of navigating to the requested host screen, wherein intermediate host screens between the current host screen and the requested host screen are unsent to a client. Applicants agree with the Examiner that *Gerra* does not teach "wherein the intermediate screen is not presented to the user" (*Office Action*, page 8). However, Applicants submit that *Kessenich* also fails to teach this feature. The Examiner refers to the following passages as evidence that *Kessenich* teaches this feature:

A system for rapidly and easily searching large collections of documents using standard web browser programs as the user interface. The present invention parses a collection of text documents to identify symbols therein and builds a database file which identifies the file and line locations of each symbol identified. The database file is constructed to permit rapid searching for symbols to permit interactive use of the present invention as a search tool. A database client process interacts with the web browser via standard CGI techniques to convert browser commands and queries into appropriate server process requests. A server process receives such requests and manipulates the database files in response to the requests. Query results returned to the client process are then reformatted by the client process to return a document with hypertext links in place of search keys located in the database (e.g., an HTML page). The system of the present invention thereby provides for rapid searching of large collections of text documents which is not coupled to a specific toolset used to create any one of the documents and which uses a simple and well-known user interface, namely: web browsers.

(*Kessenich*, Abstract).

Element 824 is operable to determine whether additional files in the collection of text documents remain to be parsed and processed as described above. If no additional files remain to be processed, processing continues at element 828 to generate the persistent storage physical format of the database file as described below with respect to FIG. 3. The above identified operations of database builder process 110 preferably construct in-memory computer data structures as well known to those skilled and art. The in memory data structures are traversed by element 828 in a sequence to produce the optimal physical storage structure described herein below. If element 824 determines that no further files require processing, element 826 is next operable to begin processing the next source document in the collection of text documents. Processing then continues by looping back to element 806 to continue processing with a new source file.

Database File Structure

FIGS. 2 and 3 depict the database file data structures in two forms. First, FIG. 2 describes the database file data structure in logical terms, as logically understood and manipulated by database server process 104 to perform requested queries. FIG. 3, by way of contrast, is the preferred physical embodiment of the database file logically depicted in FIG. 2. In particular, the preferred embodiment depicted in FIG. 3 stores the database file in such a manner as to reduce its total size and to improve performance in accessing the database file. More specifically, the preferred physical embodiment of database file 108 uses a compressed encoding of integers (as discussed herein below) to reduce storage requirements for the database file and to improve overall access performance to the database file. Furthermore, the preferred physical embodiment of the database file as depicted in FIG. 3 improves access

performance thereto by storing the data in a sequence which more closely matches the locality of references typified by database server process 104. In other words portions of the database file which are likely to be accessed chronologically near one another are stored physically near one another to reduce access times required for reading the database file.

(*Kessenich*, col. 9, lines 16-55).

FIG. 3 is a schematic representation of the preferred physical embodiment of database file 108 of the present invention. As noted above, the preferred physical embodiment of database file 108 shown in FIG. 3 utilizes compressed encoding of integer values (described herein below) to reduce the memory requirements for storage of database file 108. In addition, the organization of entries in the physical embodiment of database file 108 as depicted in FIG. 3 improves access to database file 108 by improving locality of references.

The preferred database file physical structure includes all requisite data in a preferred order as follows. First, the number of path names comprising the list of text documents is encoded as in integer value in #paths 300. Next, the actual string data of the path names of the collection of text documents is encoded in paths 302. The strings representing the file (path) names of the text documents used to construct the database are concatenated (preferably with an intervening separator character) in the order in which they were processed by the database builder process. Next, the symbols and associated file and line numbers are stored in a concatenated form for each element of the array of hash table bucket pointers (stored later in the physical format). Specifically, each bucket list entry 304 includes the symbols (as concatenated strings) in that bucket followed by the list of files and corresponding line numbers within those files where the symbols are located. Symbol and file/line index offsets 306 provide pointers into the bucket list entries 304 for each distinct symbol in the list of symbols for particular bucket list entry 304. Next, hash table chain offsets 308 provide offsets the into symbol and file/line index offsets 306 indicating the offset of the first symbol in the symbol list associated with the hash bucket pointer. Table offset 310 provides a pointer to the first hash table chain offsets 308. Lastly, table size of 312 provides the entire size of the hash table which in turn provides the starting position of symbol and file/line index offsets 306.

(*Kessenich*, col. 10, lines 25-61).

FIG. 10 is a flowchart describing methods operable within the database client process of the present invention. As noted above, a web browser invokes the services of the present invention via the database client process using the CGI communication gateway standards. The database client process, in turn, communicates with the database server process to

effectuate the query operations requested by the web browser. Those skilled in art will recognize that the features of the present invention may be implemented with or without such a client/server architecture. As noted above, a web browser (via communications with a web server process) may invoke a database manipulation program which directly accesses the database file rather than doing so through a database server process. The client/server model of the present invention provides benefits in coordinating multiple shared simultaneous access to the database file. In addition, the database client/server model preferred in the current invention permits the web browser, web server, and database server processes to be distributed over independent computing nodes. In other words, the client/server model preferred in the present invention is more easily integrated into a distributed computing environment wherein processes communicate in a standardized manner regardless of the physical computing node on which they are operating. Lastly, the database server process of the present invention, as discussed in additional detail below, supports a query command stream and returns its results essentially in ASCII text. This allows the database server process to be developed, tested, and debugged independent of the database client process.

(*Kessenich*, col. 13, lines 26-55).

FIG. 11 is a flowchart describing the processing performed by database server process 104 of the present invention. Element 1100 is first operable to receive a search command from the database client process. As noted above search commands received from the database client process are formatted in an internal format supported and defined by the database server process as discussed herein below. Element 1102 is next operable to spawn a thread for processing of the received search command. Well-known multi-threaded programming techniques are applied to permit multiple search commands to be processed on behalf of multiple database client processes. The multi-threaded programming technique also permits the server process to more easily "cleanup" on behalf of a failed processing thread. For example, failure of a single thread, processing a particular search request, does not impact concurrent processing by other threads of other search requests on behalf of other database client processes. The multi-threaded aspect of the database server processing is depicted in FIG. 11 by the multiple arrows exiting from processing of element 1102. The newly spawned thread continues processing with element 1104-1110 through to completion. The main line database server process continues processing by looping back to element 1100 to await receipt of another search request from another database client process.

The newly spawned thread of the database server process continues with element 1104 to process the search request received from the database client process. Element 1106 is next operable to determine if the query was for the contents of a file (a file contents query as generated

> by the browser program). If not, processing continue with element 1110 to transmit the search results to the database client process for further processing on behalf of the web browser program. If the query is a file content query, processing continues with element 1108 to tokenize the file content query results.
>
> As noted above, each symbol in a file content query is tokenized by the database server process. In particular, each symbol in the file content text stream is delimited by the TOKEN.sub.-- START and TOKEN.sub.-- END characters as discussed below. The tokenized results are then transmitted to the database client by operation of element 1110 for further processing on behalf of he web browser program.

(*Kessenich*, col. 14, line 36 to col. 15, line 10).

The passages fail to mention the feature of not sending the intermediate host screens between the current host screen and the requested host screen to the client. To the contrary, the passages above teach a system for enhancing a web browser wherein a user can quickly find pertinent information in a set of documents so large it cannot be printed, read, or even linearly searched interactively by a user with or without a computer. The cited portions of *Kessenich* teach building a database of search keys or symbols found in a collection of documents. The *Kessenich* system searches the database to locate a desired symbol, keyword, or file in the collection of text documents and displays the search results on the web browser display. The search results are converted to a page having hyperlinks for each search key found in the collection of text documents. The converted page with the hyperlinks is then displayed on the computer screen by the web browser. The user can then access the documents containing the desired information via the hyperlinks. As a result, an intermediate screen is sent to the client (in the form of the converted page displaying the search results) between the current host screen and the requested host screen. Thus, *Kessenich* has not been shown to teach the feature of navigating to the requested host screen, wherein intermediate host screens between the current host screen and the requested host screen are unsent to a client, and therefore fails to anticipate claim 34.

Claims 2-11, 13-22, 24-33, and 35-37 are dependent claims depending upon independent claims 1, 12, 23, and 34, respectively. Applicants have already demonstrated claims 1, 12, 23, and 34 to be in condition for allowance. Applicants

respectfully submit that claims 2-11, 13-22, 24-33, and 35-37 are also allowable, at least by virtue of their dependency on an allowable claim.

Therefore, the rejection of claims 1-4, 8, 9, 1-15, 19, 20, 22-26, 30, 31, 33, 34, 36, and 37 under 35 U.S.C. § 102 has been overcome.

Furthermore, *Gerra* does not teach, suggest, or give any incentive to make the needed changes to reach the presently claimed invention. *Gerra* actually teaches away from the presently claimed invention because it teaches displaying a link within an internet-based application in order to transfer information to a legacy application in a separate frame as opposed to a having the formatted host screen (legacy application) display selectable links to other screens within the host system as in the presently claimed invention. Absent the examiner pointing out some teaching or incentive to implement *Gerra* and having the formatted host screen (legacy application) display selectable links to other screens within the host system, one of ordinary skill in the art would not be led to modify *Gerra* to reach the present invention when the reference is examined as a whole. Absent some teaching, suggestion, or incentive to modify *Gerra* in this manner, the presently claimed invention can be reached only through an improper use of hindsight using the applicants' disclosure as a template to make the necessary changes to reach the claimed invention.

II. **35 U.S.C. § 103, Obviousness, Claims 5, 6, 16, 17, 27, and 28**

The Examiner has rejected claims 5, 6, 16, 17, 27, and 28 under 35 U.S.C. § 103 as being unpatentable over *Gerra* in view of *Sealand et al.* (US Patent No. 6,484,176). This rejection is respectfully traversed.

The Examiner bears the burden of establishing a *prima facie* case of obviousness based on the prior art when rejecting claims under 35 U.S.C. § 103. *In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992). For an invention to be prima facie obvious, the prior art must teach or suggest all claim limitations. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974).

The combination of *Gerra* and *Sealand* fail to teach or suggest the present invention as recited in claims 5, 6, 16, 17, 27, and 28. Although *Sealand* may teach a portable data processing system, (*Sealand*, Abstract; col. 2, lines 18-67; col. 3, lines 1-12;

and col. 4, lines 33-65), the *Gerra* reference still does not teach or suggest all the claim limitations in claims 5, 6, 16, 17, 27, and 28, as argued in the response to the rejection of claim 1 above. Claims 5, 6, 16, 17, 27, and 28 are patentable over the cited references because the combination of the *Gerra* reference with *Sealand* would not reach the presently claimed invention. The features relied upon as being taught in the *Gerra* reference are not taught or suggested by that reference, as explained above. As a result, a combination of these references would not reach the claimed invention in claims 5, 6, 16, 17, 27, and 28.

Therefore, the rejection of claims 5, 6, 16, 17, 27, and 28 under 35 U.S.C. § 103 has been overcome.

III. **35 U.S.C. § 103; Obviousness. Claims 7, 18, 29, and 35**

The Examiner has rejected claims 7, 18, 29, and 35 under 35 U.S.C. § 103 as being unpatentable over *Gerra* in view of *Kessenich et al.* (US Patent No. 6,055,538). This rejection is respectfully traversed.

Claim 35 was canceled. Therefore, the rejection of claim 35 under 35 U.S.C. §103 is now moot.

As for claims 7, 18, and 29, the combination of *Gerra* and *Kessenich* fail to teach or suggest the present invention as recited in claims 7, 18, and 29. The *Gerra* reference still does not teach or suggest all the claim limitations in claims 7, 18, and 29, as argued in the response to the rejection of claim 1 above. Claims 7, 18, and 29 are patentable over the cited references because the combination of the *Gerra* reference with *Kessenich* would not reach the presently claimed invention. The features relied upon as being taught in the *Gerra* reference are not taught or suggested by that reference, as explained above. As a result, a combination of these references would not reach the claimed invention in claims 7, 18, and 29.

Therefore, the rejection of claims 7, 18, and 29 under 35 U.S.C. § 103 has been overcome.

7/23/03

IV. Conclusion

It is respectfully urged that the subject application is patentable over the cited references and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: 7/23/03

Respectfully submitted,

Cathrine K. Kinslow
Reg. No. 51,886
Carstens, Yee & Cahoon, LLP
P.O. Box 802334
Dallas, TX 75380
(972) 367-2001
Attorney for Applicants